Relatório de Análise dos Algoritmos de Ordenação

Aluno: Vicente Theis Freiberger

---

1. Introdução

O presente relatório visa analisar o desempenho de três algoritmos de ordenação: Bubble Sort, Insertion Sort e Quick Sort. Esses algoritmos foram aplicados a conjuntos de dados com diferentes tamanhos e características de ordenação (aleatório, crescente e decrescente). O objetivo é observar e comparar a eficiência de cada algoritmo em cenários variados, medindo o tempo de execução em nanosegundos.

2. Métodos

Foram aplicados os algoritmos de ordenação nos seguintes conjuntos de dados:

- Aleatório: Conjuntos com dados desordenados.
- Crescente: Conjuntos com dados já em ordem crescente.
- Decrescente: Conjuntos com dados em ordem decrescente.

Cada conjunto foi testado em tamanhos de 100, 1.000 e 10.000 elementos, conforme descrito no enunciado do trabalho.

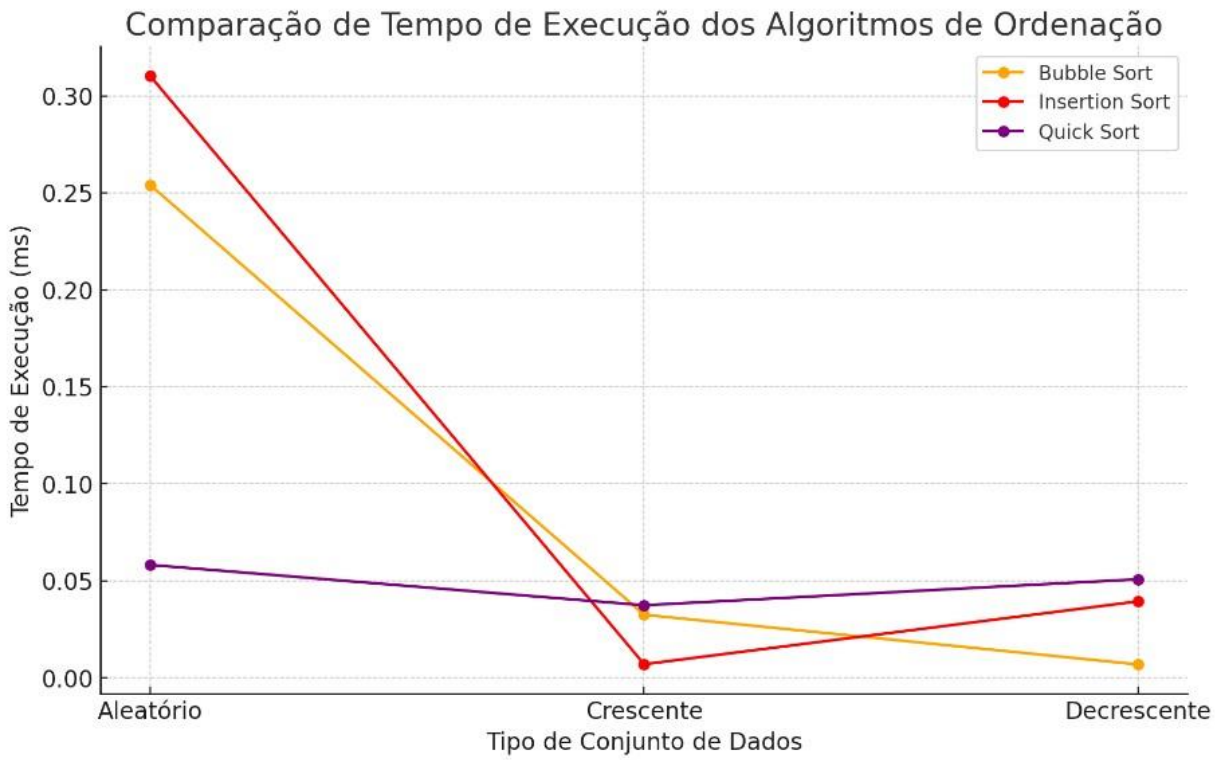
3. Resultados

Tabela 1: Tempo de Execução (em nanosegundos) dos Algoritmos de Ordenação

Conjunto de Dados	Tamanho	Bubble Sort	Insertion Sort	Quick Sort
Aleatório	100	254,100 ns	310,300 ns	58,200 ns
	1,000	N/A (arquivo não encontrado)	N/A	N/A
	10,000	248,587,500 ns	69,210,000 ns	2,308,000 ns
Crescente	100	32,500 ns	7,000 ns	37,400 ns
	1,000	4,100,200 ns	7,400 ns	1,126,900 ns

	10,000	48,276,400 ns	82,000 ns	79,898,600 ns
Decrescente	100	6,900 ns	39,400 ns	50,800 ns
	1,000	718,100 ns	2,812,500 ns	785,900 ns
Conjunto de Dados	**Tamanho**	**Bubble Sort**	**Insertion Sort**	**Quick Sort**
	10,000	83,084,200 ns	93,941,300 ns	70,422,500 ns

Tabela 1 Grafico da comparação entre tempos



4. Análise dos Resultados

Com base nos dados coletados, podemos observar o seguinte:

1. **Quick Sort** foi o algoritmo mais eficiente na maioria dos casos, especialmente para grandes volumes de dados (10.000 elementos). Isso ocorre porque o Quick Sort, em média, apresenta complexidade O(n log n), enquanto o Bubble Sort possui complexidade O(n^2).

2. **Insertion Sort** apresentou bons tempos em listas já ordenadas (crescente), o que é esperado, pois ele tem um desempenho quase linear em listas já ordenadas, com complexidade O(n) nesses casos.
3. **Bubble Sort** foi o algoritmo com pior desempenho em praticamente todos os cenários, o que é esperado devido à sua complexidade quadrática. Seu tempo de execução aumentou significativamente conforme o número de elementos crescia, especialmente em dados desordenados.

4. Em dados decrescentes, tanto o Insertion Sort quanto o Bubble Sort apresentaram tempos de execução mais altos comparados aos dados em ordem crescente, o que ocorre porque esses algoritmos fazem mais trocas ao lidar com dados em ordem inversa.

5. Conclusão

Os resultados mostram que o **Quick Sort** é o mais adequado para grandes volumes de dados e situações desordenadas. O **Insertion Sort** é eficiente para dados ordenados ou com pequenas perturbações. O **Bubble Sort**, devido ao seu alto custo computacional, é menos eficiente e adequado apenas para conjuntos de dados muito pequenos ou para fins didáticos.